NormTab: Improving Symbolic Reasoning in LLMs Through Tabular Data Normalization

Md Mahadi Hasan Nahid
University of Alberta
mnahid@ualberta.ca

Davood Rafiei
University of Alberta
drafiei@ualberta.ca

Abstract

In recent years, Large Language Models (LLMs) have demonstrated remarkable capabilities in parsing textual data and generating code. However, their performance in tasks involving tabular data, especially those requiring symbolic reasoning, faces challenges due to the structural variance and inconsistency in table cell values often found in web tables. In this paper, we introduce NormTab, a novel framework aimed at enhancing the symbolic reasoning performance of LLMs by normalizing web tables. We study table normalization as a stand-alone, one-time preprocessing step using LLMs to support symbolic reasoning on tabular data. Our experimental evaluation, conducted on challenging web table datasets such as WikiTableQuestion and TabFact, demonstrates that leveraging NormTab significantly improves symbolic reasoning performance, showcasing the importance and effectiveness of web table normalization for enhancing LLM-based symbolic reasoning tasks.

1 Introduction

Tables are a fundamental format for structured data representation and are widely used across various sources, including relational databases, web pages, and financial documents. However, many tables within documents and web pages are designed for direct human consumption and often lack the strict formatting that is expected in relational tables. This discrepancy poses significant challenges when querying them using languages such as SQL, integrating them with relational databases, and processing them within applications.

Large Language Models (LLMs) (Brown et al., 2020) have emerged as powerful tools for semantic parsing both textual and tabular data and performing complex tasks such as code generation. Trained on vast amount of Internet data, including both text and tables, and employing techniques such as Chain of Thought (CoT) prompting (Wei et al., 2022) and self-consistency (Wang et al., 2023), these models outperform many traditional models on various table reasoning tasks (Gu et al., 2022; Chen et al., 2020; Herzig et al., 2020; Wang et al., 2019). However, their performance in tasks involving tabular data, particularly those requiring symbolic reasoning, is often hindered by the structural variability and inconsistencies commonly found in web tables. Symbolic reasoning over tables necessitates a clear understanding of the table structure and values, and may involve constraining rows and columns, which can be challenging when dealing with unstructured or noisy web tables (Pourreza and Rafiei, 2023; Ni et al., 2023; Cheng et al., 2022; Zhang et al., 2023b). Our hypothesis is that normalizing ill-formatted tables can address this challenge, enabling the execution of symbolic programs (such as SQL or Python) on the tables and making reasoning tasks involving comparison, aggregation, and mathematical calculations more manageable. Moreover, normalization may enhance the explainability by allowing the tracking of the intermediate steps in reasoning.

Consider the table QA task shown in Figure 1. Retrieving answers from the table on the left using a symbolic approach such as SQL is challenging due to the irregular structure of the data and the limitations of SQL. While an LLM may handle simple look-up questions, it struggles with tasks requiring complex aggregation and arithmetic operations. However, the normalized version of the same table, shown on the right, can be easily analyzed, allowing text-to-SQL approaches to effectively obtain the answers to questions.

Existing models for table reasoning typically rely on a multi-step framework, where an LLM performs a sequence of actions such as adding columns before additional scripts are invoked to process data, retrieve cell values, or compute answers to questions (Liu et al., 2023; Wang et al., 2024; Zhang et al., 2023b). These models are of-

Findings of the Association for Computational Linguistics: EACL 2024, pages 3569–3585

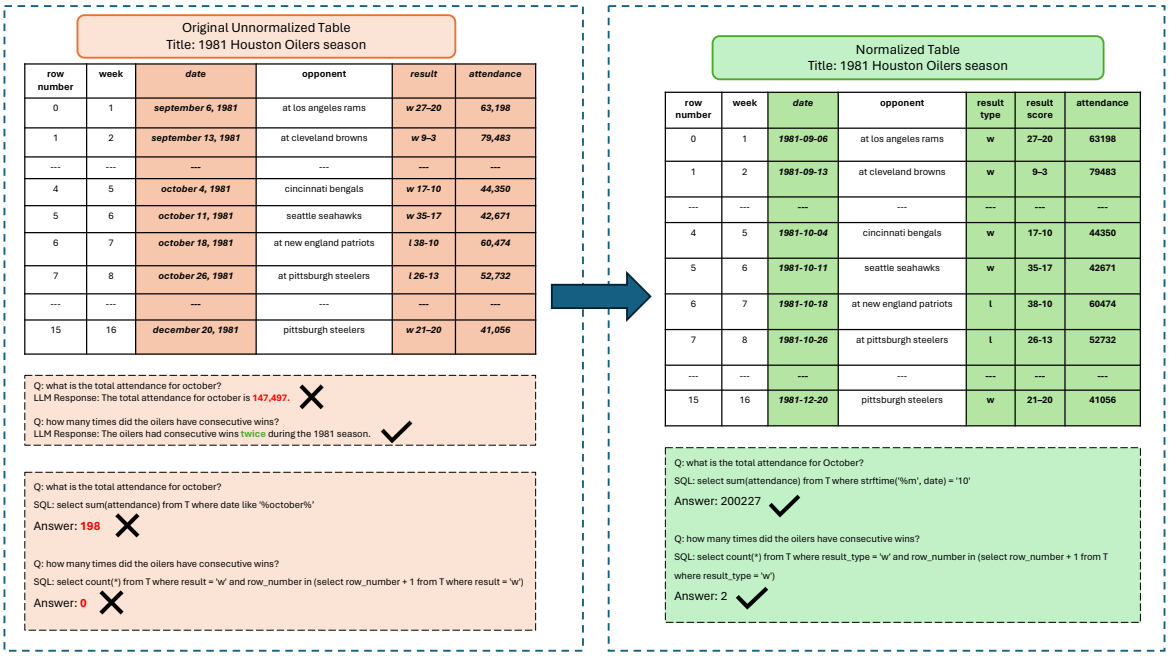

Figure 1: An example of a Table QA task, with the original unnormalized web table shown on the left and its normalized version on the right. Retrieve answers using a symbolic approach from the unnormalized table poses difficulties due to inconsistent formatting of ***date***, ***result*** and ***attendance*** columns. Also, direct querying with LLMs often fails for questions involving numerical operations. Normalization enables effective text-to-SQL conversion, as shown by the normalized table on the right.

ten dependent on question and table structure and do not address the root cause of table irregularity, making them less scalable. An alternative is normalizing tables, often part of a larger process known as data wrangling, which involves processing, cleaning and organizing data into a format that is suitable for further analysis. Significant progress has been made on data wrangling (Furche et al., 2016; Abedjan et al., 2016; Rattenbury et al., 2017), with recent approaches employing LLMs for tasks such as error detection and data imputation (Narayan et al., 2022). Selected operations, such as normalizing numbers and dates, may also be introduced into data processing pipelines to facilitate further analysis (Nahid and Rafiei, 2024). To the best of our knowledge, our work is the first to study table normalization as an stand-alone one-time preprocessing step using LLMs.

In this paper, we introduce **NormTab**, a framework designed to normalize web tables to align them with the structured format of relational database tables. NormTab addresses challenges such as structural variance, mixed data formats, and extraneous information, thereby facilitating accurate and efficient symbolic reasoning and query processing using LLMs. Our work explores two key research questions:

- **RQ1:** How can we leverage LLMs' textual understanding to effectively clean and normalize web tables?
- **RQ2:** How can web table normalization enhance table reasoning tasks, particularly in the context of LLM-based symbolic reasoning?

Our proposed solution leverages the advanced textual understanding capabilities of LLMs to independently process and normalize web tables, without relying on specific questions. By normalizing tables in this manner, we enable a robust foundation for any downstream task involving table reasoning. This approach allows for multiple questions to be asked from a single, normalized table, significantly enhancing reasoning and query capabilities. Moreover, our normalization process only needs to be performed once, unlike other models that require repeated adjustments based on different questions, highlighting a key advantage of our approach.

Through a comprehensive experimental evaluation conducted on challenging web table datasets such as WikiTableQuestions (Pasupat and Liang, 2015) and TabFact (Chen et al., 2020), we assess the effectiveness of NormTab in improving table

reasoning performance. These datasets provide diverse examples of table structures and content, allowing us to thoroughly investigate the impact of web table normalization on LLM-based symbolic reasoning tasks. By addressing RQ1 and RQ2, we aim to demonstrate the importance of web table normalization and its potential to enhance the capabilities of LLMs in handling tabular data for complex reasoning tasks.

Key Contributions of our paper are:

- We introduce NormTab, a novel framework that enhances LLMs' symbolic reasoning on tabular data by normalizing web tables. NormTab includes structure normalization (e.g., transposing tables, flattening rows and columns) and value normalization (e.g., removing extraneous strings, standardizing the formatting of dates and numbers) to ensure consistency and accuracy in reasoning tasks.
- We demonstrate how LLMs' textual understanding can be effectively utilized for data cleaning and transformation tasks, addressing challenges such as structural variance, mixed values, noise, and substring extraction in web tables
- We conduct extensive experimental evaluations using challenging web table datasets, including WikiTableQuestion and TabFact, to assess the effectiveness of NormTab in improving table reasoning performance, particularly in the context of LLM-based symbolic reasoning tasks.

2 Related Work

Our work is related to a few areas as discussed next.

General LLMs and CoT Related to our work is the line of research aimed at improving the performance of LLMs (Brown et al., 2020) on various reasoning tasks, with capabilities spanning mathematics, common sense, and symbolic reasoning (Chen, 2023; Ye et al., 2023; Cheng et al., 2022). These approaches often excel using few-shot prompts without requiring fine-tuning. Their reasoning abilities can be further enhanced by breaking complex tasks into steps, employing methods like chain-of-thought (CoT) (Wei et al., 2022) prompting and Zero-CoT. For instance, the Table-CoT (Chen, 2023) model utilizes in-context learning and CoT prompting to generate answers for table-based tasks.

Several studies have utilized instruction tuning and supervised fine-tuning to enhance the performance of LLMs on table reasoning tasks. Notable examples include TableLLaMA (Zhang et al., 2023a) and TableGPT (Zha et al., 2023), which have shown significant improvements in specific applications. In contrast, the BINDER model (Cheng et al., 2022) extends the capabilities of LLMs to programming language generation for solving commonsense problems. Additionally, the DATER approach (Ye et al., 2023) employs LLMs to decompose tables and questions, facilitating table-based QA and fact verification tasks. These diverse approaches underscore the potential of LLMs in handling complex reasoning tasks involving tabular data.

Reasoning over structured data/tables Another line of related work is reasoning over tabular data. Several studies leverage symbolic reasoning through text-to-SQL or Python code for table-based reasoning tasks. However, for effectively utilizing the symbolic code generation approach with LLMs for table reasoning tasks, it is crucial to ensure that the table is in the proper format (Pourreza and Rafiei, 2023; Rajkumar et al., 2022; Ni et al., 2023; Nahid and Rafiei, 2024; Cheng et al., 2022).

Chain-of-Table (Wang et al., 2024) enhances reasoning on tabular data by iteratively transforming and evolving table structures through a series of reasoning steps, including row/column selection, cell splitting to refine table representations for specific reasoning tasks. Their method employs in-context learning to direct LLMs in iteratively generating operations and updating the table, thus forming a chain of reasoning specific to tabular data. Liu et al. (2023) explore the capabilities of LLMs in interpreting and reasoning over tabular data, emphasizing robustness to structural perturbations, comparing textual and symbolic reasoning, and examining the potential of aggregating multiple reasoning pathways. Their findings indicate that structural variations in tables presenting the same content can significantly degrade performance, particularly in symbolic reasoning tasks. They propose a method for table structure normalization through transposition to mitigate this issue and find that while textual reasoning slightly outperforms symbolic reasoning, each approach has distinct strengths depending on the task.

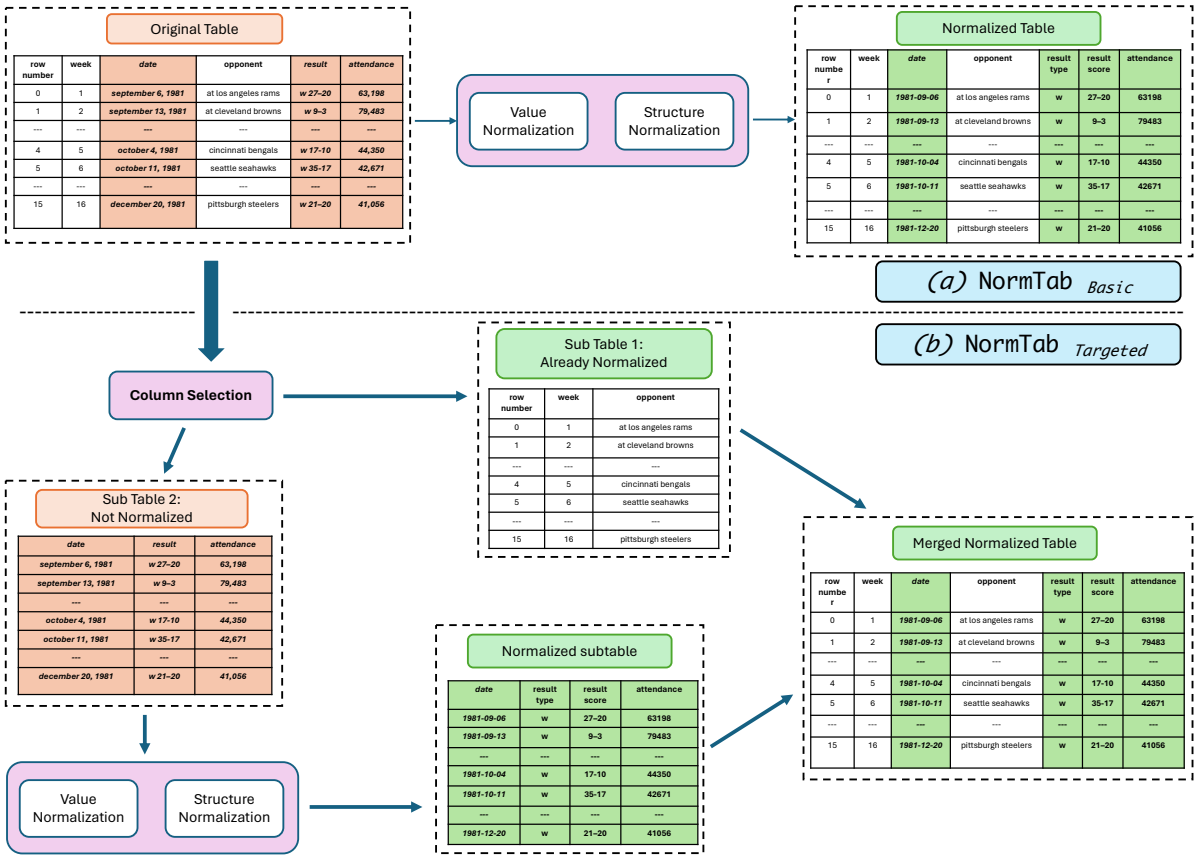

Figure 2: Overview of ***NormTab***. The methodology encompasses two distinct strategies: ***(a) Entire Table Normalization (NormTab$_{Basic}$):*** we provide the LLM with the entire web table along with specific instructions for cleaning and normalizing. The LLM reads the table and the instructions, then returns a cleaned and normalized version of the table. ***(b) Targeted Normalization (NormTab$_{Targeted}$):*** In this approach the LLM identifies and targets only the portions of the web table requiring normalization based on the table metadata and a few sample rows. The original table is split into two subtables: one for normalization and one already clean. The LLM processes the subtable that requires normalization then returned a cleaned version. Finally, the normalized subtable is merged with the clean portion, resulting in a fully cleaned and normalized table.

StructGPT (Jiang et al., 2023) employs an iterative reading-then-reasoning approach to enhance LLM reasoning for structured data, but its scalability is constrained by token limits when processing large tables. The ReAcTable model (Zhang et al., 2023b) adopts the ReAct paradigm, integrating step-by-step reasoning, external tool-based code execution, intermediate table generation, and majority voting to process tabular data. Similarly, the LEVER model (Ni et al., 2023) improves language-to-code generation by validating generated programs based on their execution results, enhancing the accuracy and reliability of table reasoning tasks.

Data wrangling and imputation Normalizing tables is a crucial aspect of the broader data wrangling process, which involves processing, cleaning, and organizing data into a format suitable for further analysis. Considerable research has focused on data wrangling, addressing challenges such as error detection, data imputation, and standardization of data formats (Furche et al., 2016; Abedjan et al., 2016; Rattenbury et al., 2017). Recent approaches have leveraged the capabilities of LLMs for these tasks. For instance, Narayan et al. (2022) demonstrated the effectiveness of LLMs in identifying errors and imputing missing data, showcasing how these models can enhance the data wrangling process. By integrating LLMs, the efficiency and accuracy of preparing data for analysis can be significantly improved, streamlining and automating many aspects of data wrangling. Operations like normalizing numbers and dates can be incorporated into data processing workflows to aid in subsequent analysis (Nahid and Rafiei, 2024).

All these works highlight the importance of table normalization in improving LLMs' performance

on tabular data, paving the way for more effective and accurate table reasoning models.

3 Methodology

Our methodology encompasses several essential parts designed to ready web tables for proficient reasoning by LLMs.

3.1 Normalization Operations

The normalization operations in **NormTab** can be divided into two groups: **(1)** value normalization and **(2)** structural normalization. The former involves splitting cells to add new columns, handling empty cells and value ranges, removing extraneous strings, and normalizing data formats such as dates and numerical values to ensure consistency and accuracy in reasoning tasks. Structural normalization, on the other hand, aims to detect structural variance by analyzing the first row and first column of a web table and determining whether a transposition is needed. If transposition is required, we address this issue by flipping the rows and columns.

Value Normalization: Our value normalization is based on the principle that every cell in a table must contain an atomic value (e.g., string, date, number), meaning that cell content cannot be composite or multi-valued. This principle, known as the first normal form in database systems (Kifer et al., 2005), ensures that cell values can be smoothly queried and updated without introducing anomalies.

The process of value normalization involves several critical steps to ensure data consistency and accuracy. First, we focus on value splitting and extraction, identifying and splitting all composite columns. This may involve adding new columns as necessary while ensuring that no existing columns are deleted. Next, we standardize date and numerical values to a uniform format, paying special attention to any additional strings such as currency symbols, units or comma that may accompany numerical values. Additionally, we normalize all "N/A" and blank values to NULL to maintain consistency throughout the dataset. In SQL, null values signify an attribute value that is not available or missing, and they are treated differently than any other values. SQL engines recognize the semantics of null values and consider this when processing queries. For columns containing value ranges, such as "2010/11" or "2015-2018", we split these into two separate columns to facilitate clearer data interpretation and processing.

An example of value normalization is shown in Figure 1. The original table presents date columns with dates in textual format, a result column combining match outcomes with scores, and an attendance column where numbers are written with commas. The value representation in the original table is more readable for humans; however, this format poses challenges for symbolic programs to process. Our normalization process converts the date to the "YYYY-MM-DD" format and attendance values to a pure numerical format by removing commas. Additionally, NormTab splits the composite result column into two separate columns: "result_type" and "result_score", thereby organizing the data more effectively for analysis. This standardization is crucial for maintaining data integrity across the table.

Structural Normalization: Tables can be organized either row-oriented or column-oriented. In a row-oriented table, each row typically represents an entity or a relationship between entities, while each column describes an attribute of the entity or relationship. Column-oriented tables, on the other hand, are stored in a transposed fashion. Most traditional databases store data in a row-oriented format, which is well-supported across relational databases.

Our structure normalization primarily focuses on addressing structural differences between tables to enhance their usability for reasoning tasks. Initially, we carefully examine the table structure to determine if the first row resembles a header, indicating the table is row-oriented and requires no structural changes. However, if the first column appears to serve as the header, we transpose the table to normalize its structure, ensuring that the layout aligns with our adopted tabular format. Additionally, web tables sometimes include aggregated rows or columns, which can pose challenges if specific rows or columns need aggregation to answer a query. We handle these aggregated rows by disregarding any information present in the last row that pertains to aggregated data, such as "total", "sum", or "average". This step prevents redundant or misleading data from affecting subsequent analyses and ensures that the table remains clean and focused on the relevant data points.

3.2 Normalization Approach: NormTab

As depicted in Figure 2, our methodology for normalizing web tables involves two distinct approaches to leverage the capabilities of LLMs for enhancing symbolic reasoning and query capabilities.

Entire Table Normalization (NormTab-Basic): In the first approach, we provide the LLM with the entire table along with specific instructions for cleaning and normalizing. The LLM reads the table and the instructions, then returns a cleaned and normalized version of the table. However, we observed that many web tables contain portions already in a well-structured form, with only a few columns requiring normalization. To optimize this process, we developed a modified approach.

Targeted Normalization(NormTab-Targeted): To improve efficiency, we developed a modified approach that targets only the portions of the table requiring normalization. Our analysis of web tables revealed that often only a few columns need the normalization process. This realization led to a more optimized methodology. In this more refined approach, we first ask the LLM to identify which columns require normalization and cleaning, based on the table metadata (such as column headers and titles) and a few sample rows. Once these columns are identified, we split the original table into two subtables: one that requires normalization and cleaning, and one that is already normalized and clean. We then send only the subtable that needs normalization to the LLM along with the instructions. The LLM processes this subtable and returns a cleaned and normalized version. After normalization, we merge the normalized subtable with the already clean portion of the table. This approach not only improves the efficiency of the normalization task by reducing the amount of data sent to the LLM but also ensures that the resulting table is in a consistent and accurate format suitable for subsequent reasoning and querying tasks.

Following this, we analyze the overall structure of the merged table. With the assistance of the LLM, we determine whether the table needs to be transposed based on its layout. If needed, table transposition is performed outside of the LLM. Additionally, we check if the last row contains summarized or aggregated values and if so, NormTab ignore this row. This selective column normalization method reduces the workload on the LLM, enhances efficiency, and ensures that only the necessary parts of the table are processed, thereby preserving the integrity of already structured data.

4 Experimental Setup

4.1 Dataset

We conduct experimental evaluations using two challenging web table datasets: WikiTableQuestion (WikiTQ)(Pasupat and Liang, 2015) and TabFact (Chen et al., 2020). These datasets are specifically curated to test the reasoning capabilities of models on complex tabular data. WikiTQ comprises tables extracted from Wikipedia along with corresponding natural language questions, while TabFact consists of tables sourced from Wikipedia paired with textual facts. These datasets provide a diverse range of table structures and content, allowing us to thoroughly evaluate the performance of NormTab in enhancing table reasoning tasks.

The WikiTQ standard test set comprises 416 unique tables and 4,344 samples, while the TabFact standard test set includes 298 unique tables with 2,003 samples. By utilizing these datasets, we aim to demonstrate the effectiveness of web table normalization in improving the symbolic reasoning performance of LLMs, thereby highlighting the importance of addressing the challenges posed by web table irregularities.

4.2 Baselines and Evaluation Metrics

We compare our approach with several robust baseline methods, including TableCoT (Chen, 2023), BINDER (Cheng et al., 2022), DATER (Ye et al., 2023), StructGPT (Jiang et al., 2023), ReAcTable (Zhang et al., 2023b), Rethinking-Tab-Data (Liu et al., 2023), TabSQLify (Nahid and Rafiei, 2024), and Chain-of-Table (Wang et al., 2024).

For the WikiTQ dataset, exact match (EM) accuracy was used to check if the predicted answers matched the correct ones. To address varying text formats, a pre-matching check using LLMs was incorporated (Cheng et al., 2022). The accuracy for TabFact was assessed using binary classification accuracy.

4.3 Implementation

We utilized gpt-3.5-turbo-0125 as the Language Model which supports 16k context window. We

were inspired by the prompting style from (Liu et al., 2023; Nahid and Rafiei, 2024) in our implementation of NormTab. To compare performance, we employ few-shot in-context learning. This involves supplying the LLM with the table title, table header, question, and three example rows of the table, along with the question, to generate an SQL query. The SQL query is then executed on the table to obtain the answer. Further details can be found in Appendix C, and all our code and prompts are available at https://github.com/mahadi-nahid/NormTab.

5 Results

In this section, we analyzed the performance of NormTab. To evaluate its impact, we conducted few-shot in-context learning experiments to generate SQL queries for answering specific questions. First, we performed experiments on unnormalized tables without any modifications. Then, we compared the performance on normalized tables. Additionally, we reported the performance of different normalization processes.

5.1 Results on Downstream Tasks

Table 1 and Table 2 presents a comparison between the performance of NormTab and the other baselines on WikiTQ and TabFact datasets.

In the WikiTQ dataset, the results showed that after applying the targeted version of NormTab, we achieved 61.2% accuracy, surpassing the performance of other baseline models. The targeted NormTab approach performs slightly better than the basic version, where the entire table is passed to the LLMs. This suggests that LLMs may be more effective at normalization tasks when dealing with targeted smaller tables. Additionally, we gained about 10% improvement compared to the Text-to-SQL (Rajkumar et al., 2022) model and SQL (gpt-3.5-turbo) model. Notably, Rethinking-Tab-Data (Liu et al., 2023) achieved an accuracy of 56.87% by addressing structural varience using LLMs and a Python agent. Chain-of-Table (Wang et al., 2024) employed an iterative sequence of operations to tailor complex tables to specific questions, achieving 59.94% accuracy. However, these and other baseline models are question-dependent. In contrast, our model adopts a straightforward and simple approach: it normalizes the table only once, irrespective of the question, enabling answers to be derived from the normalized table using program-aided symbolic reasoning.

Model	Acc (%)
TableCoT (Chen, 2023)	52.40
BINDER	56.74
DATER	52.80
ReAcTable	52.40
Rethinking-Tab-Data	56.87
Chain-of-Table	59.94
Text-to-SQL (Rajkumar et al., 2022)	52.90
Text-to-SQL (gpt-3.5-turbo)	51.30
NormTab$_{Basic}$ + SQL (ours)	**60.80**
NormTab$_{Targeted}$ + SQL (ours)	**61.20**

Table 1: Performance comparison of NormTab on WikiTQ dataset. The results clearly demonstrate that NormTab significantly surpasses other models in accuracy when employing symbolic reasoning.

In Table 2, we can observe a similar performance enhancement compared to the original table in table-based fact verification tasks. We achieved approximately a 6% performance improvement compared to the results of Text-to-SQL on the original table. It is worth noting that table-based fact verification differs from table-based question answering tasks. Generating a SQL query to verify a fact is more complex than simply retrieving an answer from the table. Although other models not employing program-aided symbolic reasoning perform better in this task, these models utilize LLMs for the verification task providing the whole table to the model. Our experimental results show promise for utilizing symbolic reasoning in such scenarios.

Model	Acc (%)
TableCoT-chatgpt	73.10
BINDER	79.17
DATER	78.01
Chain-of-Table	80.20
ReAcTable	73.10
Text-to-SQL (Rajkumar et al., 2022)	64.71
Text-to-SQL (gpt-3.5-turbo)	62.32
NormTab$_{Basic}$ + SQL (ours)	**67.10**
NormTab$_{Targeted}$ + SQL (ours)	**68.90**

Table 2: Performance comparison of NormTab on TabFact dataset with other models.

In this study, we also evaluated the effectiveness of our method using Gemini-1.5-flash and GPT-4-turbo. The results showed a improvement on

the WikiTableQuestions dataset, demonstrating that our model's performance is not heavily dependent on specific language models (see Table 3).

Model	Acc (WTQ)
NormTab$_{Basic}$ (gemini-1.5-flash)	61.36
NormTab$_{Targeted}$ (gemini-1.5-flash)	61.24
NormTab$_{Basic}$ (gpt-4-turbo)	61.57
NormTab$_{Targeted}$ (gpt-4-turbo)	**62.28**

Table 3: Performance of NormTab on WikiTQ dataset using Gemini-1.5-flash and GPT-4-turbo.

5.2 NormTab Evaluation

To assess the accuracy of various normalization operations, we evaluated the performance on 100 tables, with 50 tables from each dataset, WikiTQ and TabFact. Table 4 summarizes the accuracy of different normalization processes. NormTab demonstrated strong performance in normalizing dates and numbers, detecting transposition requirements, and handling aggregated summaries in the last row effectively. However, NormTab faced difficulties in extracting and cleaning values in certain critical tables where value extraction from the original table was particularly challenging. The column selection accuracy indicates that LLMs can be very effective in identifying columns where values are not in the proper format. However, the accuracy of splitting columns was low. Additional errors included managing value cleaning and handling "n/a" values. Although these tasks are challenging, the performance in these areas shows the potential for utilizing LLMs to address these tasks effectively.

Type	Accuracy
Columns Selection	91.0%
Transpose Detection	97.0%
Last Row Aggregation	100.0%
Split Column	87.0%
Date and Number	100.0%
N/A value	93.0%
Value Cleaning	82.0%

Table 4: Accuracy of ***NormTab*** in various types of normalization.

NormTab has shown superior performance compared to several robust models, demonstrating its efficacy in table normalization. A key advantage of NormTab is its use of program-aided symbolic reasoning, which streamlines code generation without requiring the entire table to be passed to the LLM. This enhances efficiency and eliminates dependencies on table size and answer position. With NormTab, only key elements like the title, header, and a few example rows are needed to generate SQL queries and obtain accurate answers. This approach reduces computational overhead while maintaining high accuracy, highlighting its practical utility in various table-based tasks.

Our normalization method, NormTab, can be beneficial for a variety of table reasoning tasks, especially those employing symbolic methods. For the same reason, we have integrated NormTab with a recent table reasoning method TabSQLify (Nahid and Rafiei, 2024), which utilizes symbolic techniques. Our evaluation reveals that integrating NormTab with TabSQLify leads to a notable 4% improvement in performance. This demonstrates NormTab's potential to enhance other symbolic frameworks by serving as an effective preprocessing step. The results is summarized in Table 5.

Model	Acc (WTQ)
TabSQLify	64.7
NormTab+TabSQLify	**68.63**

Table 5: Performance of TabSQLify integrated with NormTab on the WikiTableQuestions dataset (gpt-3.5-turbo).

In our work, each table is normalized once, as a preprocessing step, irrespective of the number and the type of questions asked. Our approach does not depend on the specific question or task. While using Self-Consistency (Wang et al., 2023; Liu et al., 2023) might seem beneficial, it requires generating multiple responses per question, which can increase costs and reduce efficiency. For instance, if we need answers to 10 questions and apply Self-Consistency with 6 paths, we end up with 60 samples (10 questions × 6 paths). Additionally, we need to send the entire table with each question, resulting in a higher number of tokens. In contrast, our method requires just a few prompts to normalize the table initially. After normalization, we do not need to pass the entire table to generate the SQL query for each question; only table metadata such as the table title, column names, and a few example rows, is needed. This one-time prepossessing step significantly reduces the overall number of tokens passed to the LLM, potentially lowering costs compared to using Self-Consistency.

5.3 Analysis

We conducted a detailed analysis of the impact of NormTab on the WikiTQ dataset. Table 6 shows that in **67%** of cases (Category A), performance improved after applying NormTab. In 24% of cases (Category B), performance remained unchanged, indicating no improvement. Additionally, in 9% of cases (Category C), performance actually decreased. The detailed experimental findings are summarized in Table 7.

Categories	Description	% of Tables
A	Where performance enhanced after applying NormTab	**67%**
B	Where no change in performance after applying NormTab	24%
C	Where the performance decreased after applying NormTab	9%

Table 6: Categories of tables on WikiTQ test dataset.

-	Tables (A)	Tables (B,C)	Overall (A,B,C)
Original	46.28%	59.62%	51.30%
NormTab	**62.55%**	56.76%	**61.20%**
Change	**+16.27**	-2.86	**+9.9**

Table 7: Result breakdown on WikiTQ dataset.

Table 7 demonstrates that NormTab can improve overall performance by 9.9%. Notably, in Category A, we observed a substantial enhancement of 16.27%. However, Categories B and C saw a slight decline in performance due to highly complex table values and structures.

The basic NormTab approach involves only one LLM call, but it requires passing the entire table to the language model as a prompt. This means the LLM must process a larger number of tokens, which can impact the overall normalization performance. Research indicates that more tokens can increase the likelihood of hallucination and can be more costly (Chen, 2023; Ye et al., 2023; Ji et al., 2023). In contrast, the targeted NormTab approach first filters out the parts of the table that are already well-formatted, thereby reducing the number of tokens sent to the LLM by focusing only on the columns that need normalization. For example, consider a table with 15 rows and 8 columns, resulting in a total of 15 * 8 = 120 table cells. In the basic NormTab approach, all 120 cells are sent to the language model along with the instructions. However, if we identify that only 5 out of 8 columns require normalization, we only need to send 15 * 5 = 75 table cells. This reduction translates to 45 fewer table cells, which is a 37.5% reduction in table size. Table 8 illustrates that we can achieve a 72% reduction in table size for both datasets by employing the targeted NormTab approach, which is quite substantial.

Dataset	NormTab (Basic)	NormTab (Targeted)	Reduction
WikiTQ	152.26	41.82	**72.53%**
TabFact	106.19	29.11	**72.58%**

Table 8: Table cell reduction in NormTab-Targeted compared to NormTab-Basic

Although the targeted NormTab approach requires additional LLM calls for tasks such as column selection and transposition detection, the total number of tokens processed is still lower than in the basic NormTab approach. While the basic approach may be suitable for smaller tables, the reduction in table size is crucial for normalizing larger tables effectively. The targeted strategy involves multiple queries, but it refines the normalization process by concentrating on specific subtasks, which may help reduce hallucination and errors. While the performance improvement appears marginal, the significant token size reduction makes the targeted NormTab approach highly beneficial for larger tables.

6 Conclusion

In conclusion, our study introduces NormTab, a framework aimed at enhancing LLMs' performance on tabular data by normalizing web tables. Through our investigation, we have shown the significance of web table normalization in overcoming challenges such as mixed values and structural variance. By leveraging LLMs' textual understanding in data cleaning and normalization, NormTab improves table reasoning. Our experiments on challenging datasets demonstrate its effectiveness. Our work contributes to advancing techniques for LLMs in handling tabular data, emphasizing the importance of addressing web table challenges for improved performance. Further research can explore additional normalization strategies and extend NormTab's applicability across various domains. This would establish a robust foundation for a wide range of downstream tasks involving table reasoning.

Limitations

Despite the advancements brought by NormTab, there are several limitations. First, while our framework significantly enhances the symbolic reasoning capabilities of LLMs on tabular data, there remains room for improvement in the normalization process, particularly with more complex table structures. Additionally, for larger tables, LLMs may sometimes produce hallucinated results, leading to inaccuracies in the normalized output, indicating a need for better handling of extensive datasets. Moreover, when working with tables containing highly noisy data, LLMs often struggle to clean and normalize the information effectively, and may generate output in an incorrect format, making it challenging to parse. The presence of excessive noise and inconsistencies can hinder the normalization process and negatively impact overall performance. Addressing these limitations is crucial for further enhancing the robustness and reliability of NormTab. As we measure the accuracy using the results obtained from LLM based Text-to-SQL model, it is important to note that some questions in the dataset may not directly map to SQL queries which may affect the performance.

Acknowledgements

We extend our sincere gratitude to all reviewers for their invaluable feedback, insightful suggestions, and positive remarks about our work. This research has been supported by the Natural Sciences and Engineering Research Council of Canada. Also, Md Mahadi Hasan Nahid was supported by the Alberta Innovates Graduate Student Scholarship.

Ethical Considerations

The datasets used in this study are accessible through the peer-reviewed articles cited in the references section. Additionally, our source code is openly available for future research under the MIT License. It is important to mention that our framework relies on GPT-3.5-turbo, which may inherit ethical concerns associated with GPT models. These concerns include the potential for generating responses to toxic content or displaying biased behavior.

References

Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting data errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment*, 9(12):993–1004.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2022. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Tim Furche, Georg Gottlob, Leonid Libkin, Giorgio Orsi, and Norman W Paton. 2016. Data wrangling for big data: Challenges and opportunities. In *19th International Conference on Extending Database Technology*, pages 473–478.

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. PASTA: Table-operations aware fact verification via sentence-table cloze pre-training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4971–4983, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Michael Kifer, Arthur Bernstein, and Philip M Lewis. 2005. Database systems: An application-oriented approach.

Tianyang Liu, Fei Wang, and Muhao Chen. 2023. Rethinking tabular data understanding with large language models. *arXiv preprint arXiv:2312.16702*.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. TabSQLify: Enhancing reasoning capabilities of LLMs through table decomposition. In *2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *Proceedings of the VLDB Endowment*, 16(4):738–746.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I Wang, and Xi Victoria Lin. 2023. Lever: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Tye Rattenbury, Joseph M Hellerstein, Jeffrey Heer, Sean Kandel, and Connor Carreras. 2017. *Principles of data wrangling: Practical techniques for data preparation*. " O'Reilly Media, Inc.".

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3774–3785, Hong Kong, China. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, et al. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *arXiv preprint arXiv:2307.08674*.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023a. Tablellama: Towards open large generalist models for tables. *Preprint*, arXiv:2311.09206.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2023b. Reactable: Enhancing react for table question answering. *arXiv preprint arXiv:2310.00815*.

A Web Tables vs Regular DB Table

There are some key differences between a web table and relational database table. Web tables and relational database tables exhibit distinct characteristics, each with its own advantages and challenges. Web tables, often sourced from online sources such as websites and spreadsheets, tend to be diverse in structure and content. They may contain varying formats, including column-based, row-based, and aggregated summaries, with irregularities such as mixed values and noisy data being common. Additionally, web tables frequently present challenges in terms of structural variance, as they may lack standardized schema and exhibit inconsistencies in data organization. In contrast, relational database tables (RDBMS) adhere to a structured schema. These tables are typically designed for efficient storage and retrieval of data, with clear relationships defined between entities through keys and constraints. Relational database tables offer advantages in terms of data integrity, consistency, and

scalability, as they enforce normalization principles and allow for efficient querying through SQL.

However, Web Tables may lack the flexibility and adaptability required to handle the diverse and unstructured nature of web data. While relational database tables excel in maintaining structured data integrity, web tables present challenges related to variability and noise, necessitating specialized techniques for effective processing and analysis.

B Comparison with Other Models

Chain-of-Table (Wang et al., 2024) enhances reasoning on tabular data by iteratively transforming and evolving table structures through a series of reasoning steps, including row/column selection and cell splitting. Their method uses in-context learning to guide LLMs in generating operations and updating the table, forming a reasoning chain specific to tabular data. Liu et al. (2023) explore LLM capabilities in interpreting and reasoning over tabular data, emphasizing robustness to structural changes and comparing textual and symbolic reasoning. They find that structural variations can significantly degrade performance, especially in symbolic reasoning tasks, and propose table structure normalization through transposition to mitigate this issue. Their study concludes that while textual reasoning slightly outperforms symbolic reasoning, both have distinct strengths depending on the task.

The ReAcTable model (Zhang et al., 2023b) follows the ReAct paradigm, incorporating step-by-step reasoning, external tool-based code execution, intermediate table generation, and majority voting to process tabular data. Recent approaches leverage LLM capabilities for these tasks. For instance, Narayan et al. (2022) demonstrated LLM effectiveness in identifying errors and imputing missing data, enhancing the data wrangling process. Integrating LLMs can significantly improve the efficiency and accuracy of preparing data for analysis, streamlining and automating many aspects of data wrangling. Operations like normalizing numbers and dates can be incorporated into data processing workflows to aid subsequent analysis (Nahid and Rafiei, 2024; Cheng et al., 2022).

Existing models and methods typically rely on multi-step frameworks where LLMs select actions, such as adding columns, and additional scripts process values based on specific questions (Liu et al., 2023; Wang et al., 2024; Zhang et al., 2023b). However, these approaches are question-dependent and do not comprehensively address the root issue of table normalization. NormTab differs by focusing on normalizing tables once, regardless of the question, allowing answers to be derived from the normalized table using program-aided symbolic reasoning. This approach reduces dependencies on table size and answer position, enhancing efficiency and versatility in table reasoning tasks.

C Implementation Settings

The dataset we used contains only the gold answers and lacks the original SQL queries needed to extract these answers. Additionally, the dataset does not include normalized or cleaned versions of the tables. Data contamination can indeed impact methods where the question and table are provided to the LLM, which might then rely on knowing the gold answer directly. In our approach, however, we focus on converting tables into a normalized format. Since the dataset does not provide such cleaned versions, data contamination does not affect our method. Moreover, our approach generates SQL queries based on the question and table metadata, rather than relying on pre-existing gold SQL queries. We obtain the answers by executing these generated SQL queries. Therefore, data contamination does not impact our method.

Our implementation of NormTab was inspired by the prompting techniques used in (Liu et al., 2023; Nahid and Rafiei, 2024). We configured the in-context learning hyperparameters for gpt-3.5-turbo-0125 according to the specifications outlined in Table 9.

Parameter	Coll Selection	Transpose Detection	NormTab
temperature	0.3	0.3	0.7
top_p	1	1	1
sample_n	1	1	1
max_tokens	100	100	4500
num_shots	6	1	1

Table 9: The hyper-parameters we set in NormTab

D Example Prompts

The prompt used in NormTab is described in the following Figures.

Column Selection Prompt

```
You are an advanced AI capable of analyzing and understanding information within tables.
Your task is to normalize a web table so that it can be converted as a relational database table.

### Instructions: Identify the columns based on the following instructions

1. Identify the columns If some of the values of a column needed to be extracted then extract the string and add it in new
columns.
2. Identify the columns that has date type value and the numerical value.
3. Identify the columns that has numerical values containing extra string such as '$' or units.
4. Identify the columns that has 'N/A',  blank or null.
5. Identify the columns that contain ranges  such as (20-2), 2010/11, 2015-2018 etc.

### Task: Your task is to identify which columns needed to be  normalized to convert this table as a regular normalized
relational database table so that we can run sqlite sql query over this table.

### Table:
Read the table below regarding "2008 Clásica de San Sebastián"

rank | cyclist | team | time | uci_protour_points
1 | alejandro valverde (esp) | caisse d'epargne | 5h 29' 10" | 40
2 | alexandr kolobnev (rus) | team csc saxo bank | s.t. | 30
3 | davide rebellin (ita) | gerolsteiner | s.t. | 25

Table Coll: (rank, cyclist, team, time, uci_protour_points)

### Response: normalize_coll = ['cyclist']

### Table:
Read the table below regarding "Sky Track Cycling"

date | competition | location | country | event | placing | rider | nationality
31 october 2008 | 2008–09 world cup | manchester | united kingdom | sprint | 1 | victoria pendleton | gbr
31 october 2008 | 2008–09 world cup | manchester | united kingdom | keirin | 2 | jason kenny | gbr
1 november 2008 | 2008–09 world cup | manchester | united kingdom | sprint | 1 | jason kenny | gbr

Table Coll: (date, competition, location, country, event, placing, rider, nationality)

### Response: normalize_coll = ['date', 'competition']
--- --- --- --- ---
--- --- --- --- ---
### Table:
Read the table below regarding "[TITLE]"

[3 ROWS OF THE TABLE]

### Output: Let's think step by step, and generate the final output based on the instructions without any explanation. Ensure
the final output is only "normalize_coll = [col1, col2, col3,…]"form, no other form.
### Response:
```

Figure 3: Column Selection prompt.

Summarized Last Row Detection Prompt

You are an advanced AI capable of analyzing and understanding information within tables.

Task: You are given the last row of a table. Your task is to detect if the last row has any information like aggregated rows such as ‘total’, ‘sum’ or 'average’.

Last row: ***[LAST ROW AS A LIST]***

Directly give your choice. Ensure the format is only "YES or NO" form, no other form, without any explanation.

Response:

Transpose Detection Prompt

You are an advanced AI capable of analyzing and understanding information within tables.

Read the first 3 rows of the table regrading ***"[TITLE]"***

Table:

[3 ROWS OF THE TABLE]

Headings of a table are labels or titles given to rows or columns to provide a brief description of the data they contain. Based on the given table, the headings of the table are more likely to be:

(A): ***[FIRST ROW AS LIST]***

(B): ***[FIRST COLUMN AS LIST]***

Directly give your choice. Ensure the format is only "(A) or (B)" form, no other form, without any explanation.

Response:

Figure 4: Summarized last row detection and transpose detection prompt.

NormTab Prompt

```
You are an advanced AI capable of analyzing and understanding information within tables.
Your task is to normalize a web table and convert it into a relational database table, enabling the execution of SQL queries on
the data.

Read the table below regarding "[TITLE]"

### Table:


[TABLE]


### Task: Your task is to normalize the structure and the values of each cell to convert this table as a regular normalized
relational database table so that we can run sqlite sql query over this table.


### Instructions:

1. If some of the values needed to be splitted or extracted then extract the string and add it in new columns. i.e. from
'alejandro valverde (esp)' country 'esp' can be extracted and added to the new column.
2. Make sure the date and the numerical value is normalized to a uniform format. The date format should be (YYYY-MM-DD).
3. Be cautious of numerical values that contain comma or any extra string such as '$', '%' or units.
4. Convert the 'N/A' or null values to blank.
5. Handle the columns that contain ranges such as 2010/11, 2015-2018 etc to  two separate columns.
6. Never delete any columns or rows.
7. Carefully remove extraneous characters if needed.



### Output: Let's think step by step and generate the final output table based on the instructions without any explanation.
Ensure the final output is only "normalized_table = [[col1, col2, col3,…], [row11, row 12, row13,.....],....]"form, no other
form.


### Response:
```

Figure 5: NormTab Instruction prompt.

```
Generate SQL with no explanation given the question and table to answer the question correctly.

### SQLite table properties:

Table: Marek Plawgo(row_number,year,competition,venue,position,event,notes)

3 example rows:
 select * from T limit 3;
row_number | year | competition | venue | position | event | notes
0 | 1999 | european junior championships | riga, latvia | 4th | 400 m hurdles | 52.17
1 | 2000 | world junior championships | santiago, chile | 1st | 400 m hurdles | 49.23
2 | 2001 | world championships | edmonton, canada | 18th | 400 m hurdles | 49.8

Q: when was his first 1st place record?
SQL: select year from T where position = '1st' order by year asc limit 1

---- ----- ------ ----- ----
---- ----- ------ ----- ----
---- ----- ------ ----- ----

### SQLite table properties:

Table: Figure skating at the Asian Winter Games(row_number, rank, nation, gold, silver, bronze, total)

3 example rows:
 select * from T limit 3;
 row_number | rank | nation | gold | silver | bronze | total
0 | 1 | china | 13 | 9 | 13 | 35
1 | 2 | japan | 7 | 10 | 7 | 24
2 | 3 | uzbekistan | 1 | 2 | 3 | 6

Q: what is the average number of gold medals won by china, japan, and north korea?
SQL: select avg(gold) from T where nation in ('china', 'japan', 'north korea')

### SQLite table properties:

Table: [TITLE] ([COLUMN NAMES])

3 example rows:
 select * from T limit 3;

 [THREE EXAMPLE ROWS]

Q: [QUESTION]
SQL:
```

Text-to-SQL Prompt template (WikiTQ)

Figure 6: Text-to-SQL prompt template for the Table QA Task on the WikiTQ dataset.

Text-to-SQL Prompt template (TabFac)

```
Generate SQL given the statement and table to verify the statement correctly.

### SQLite table properties:

Table: 1947 kentucky wildcats football team(row_number, game, date, opponent, result, wildcats_points, opponents, record)

3 example rows:
 select * from T limit 3;
row_number | game | date | opponent | result | wildcats_points | opponents | record
0 | 1 | sept 20 | ole miss | loss | 7 | 14 | 0 - 1
1 | 2 | sept 27 | cincinnati | win | 20 | 0 | 1 - 1
2 | 3 | oct 4 | xavier | win | 20 | 7 | 2 - 1

Q: the wildcats kept the opposing team scoreless in four games
SQL: SELECT (SELECT COUNT(*) FROM T WHERE opponents = 0) = 4

---- ----- ------ ----- ----
---- ----- ------ ----- ----
---- ----- ------ ----- ----

### SQLite table properties:

Table: katsuya inoue(row_number, res, record, opponent, method, event, round, time, location)

3 example rows:
 select * from T limit 3;
row_number | res | record | opponent | method | event | round | time | location
0 | loss | 19 - 9 - 4 | naoyuki kotani | submission (armbar) | pancrase - impressive tour 9 | 1 | 1:44 | tokyo , japan
1 | loss | 19 - 8 - 4 | kota okazawa | ko (punch) | pancrase - impressive tour 4 | 1 | 2:42 | tokyo , japan
2 | win | 19 - 7 - 4 | katsuhiko nagata | decision (unanimous) | gcm - cage force 17 | 3 | 5:0 | tokyo , japan

Q: in tokyo , japan , hikaru sato 's match ended before round 2
SQL: SELECT (SELECT COUNT(*) FROM T WHERE location = 'tokyo , japan' AND round < 2) > 0;

### SQLite table properties:

Table: [TITLE] ([COLUMN NAMES])

3 example rows:
 select * from T limit 3;

 [THREE EXAMPLE ROWS]

Q: [STATEMENT]
SQL:
```

Figure 7: Text-to-SQL prompt template for the Table-based Fact Verification Task on the TabFact dataset.